BUT 1 - Projet Java

Crazy Circus

Université Paris Cité

Nicolas Tassin - Florian Colin - Victor Degrolard

Table des matières

Présentation de l'application 3
Rôle fonctionnel 3
Description des classes 3
Les entrées / sorties 5
Les recettes 6
Organisation des tests 6
Diagramme UML 6
Bilan 7
Points à améliorer 7
Points positifs 7
Conclusion 7

Présentation de l’application

Rôle fonctionnel

L'application Crazy Circus est une simulation d'un jeu de société pour deux joueurs, dans lequel chaque joueur possède deux podiums, l'un bleu et l'autre rouge. Les podiums contiennent des animaux représentés par des cartes, et les joueurs doivent effectuer des actions pour déplacer les cartes d'un podium à l'autre. Le but du jeu est d'avoir le plus de points à la fin de la partie, en obtenant des points pour chaque animal de sa couleur placé sur le podium adverse. L'application gère les déplacements des cartes entre les podiums et calcule les scores pour les joueurs.

Description des classes

Classe Animaux

La classe Animaux représente les animaux du cirque. Elle contient les attributs suivants :

- id : identifiant unique de l'animal
- name : nom de l'animal
- species : espèce de l'animal
- trick : tour que l'animal peut effectuer

Elle fournit également des méthodes pour accéder et modifier ces attributs, ainsi qu'une méthode pour obtenir une représentation sous forme de chaîne de caractères de l'animal.

Classe Map

La classe Map représente la carte du cirque, qui contient deux podiums (un podium rouge et un podium bleu). Elle contient les attributs suivants :

- blue : podium bleu
- red : podium rouge

Elle fournit des méthodes pour accéder et modifier ces attributs.

Classe Order

La classe Order représente les ordres que le public peut donner aux animaux. Elle contient plusieurs méthodes statiques qui permettent de réaliser les différentes actions possibles :

- KI : déplace le dernier animal du podium bleu vers le podium rouge
- LO : déplace le dernier animal du podium rouge vers le podium bleu
- SO : échange le dernier animal du podium rouge avec le dernier animal du podium bleu
- NI : déplace tous les animaux du podium bleu d'une position vers la droite (le dernier animal devient le premier)

- MA : déplace tous les animaux du podium rouge d'une position vers la gauche (le premier animal devient le dernier)

La classe Order utilise également la classe Podium pour réaliser ces actions.

Classe Player

La classe Player représente un joueur du jeu. Elle contient les attributs suivants :

- name : nom du joueur
- canPlay : indique si le joueur peut jouer ou non
- score : score du joueur

Elle fournit des méthodes pour accéder et modifier ces attributs, ainsi qu'une méthode pour ajouter 1 au score du joueur.

Classe Podium

La classe Podium représente un podium du cirque, qui contient une liste d'animaux. Elle contient les attributs suivants :

- container : liste des animaux du podium
- name : nom du podium
- color : couleur du podium (rouge ou bleu)

Elle fournit des méthodes pour accéder et modifier ces attributs, ainsi que des méthodes pour ajouter, enlever et déplacer les animaux du podium. Elle contient également une méthode pour comparer le contenu de deux podiums.

Les entrées / sorties

Les entrées sorties de l'application

Entrée [Type INT] : nombre de joueurs

Sortie attendue : « Joueur x comment t'appelle-tu » répété x fois (x étant le nombre de joueurs)

Sortie possible : « Vous devez renseigner un chiffre »

Entrée [Type String] : nom du joueur

Sortie attendue : « Bonjour, Joueur x : String ! »

Sortie possible : « Vous devez renseigner votre pseudo »

Entrée [Type String] : Nom du joueur & action

Sortie attendue : Mise à jour du plateau de jeu

Sortie possible : « Tu as déjà joué cette manche »

Sortie possible : « Dommage pour cette manche, plus personne ne peut jouer ! »

Sortie possible : « Joueur inconnu »

Quand une manche se termine la console peut renvoyer :

« Joueur remporte cette manche SCORE point(s)

« Manche suivante ! (Encore NB_POSSIBILITE manche(s)) »

Quand une partie se termine la console peut renvoyer :

« Fin de partie 😊 »

« Aucun gagnant pour cette fois »

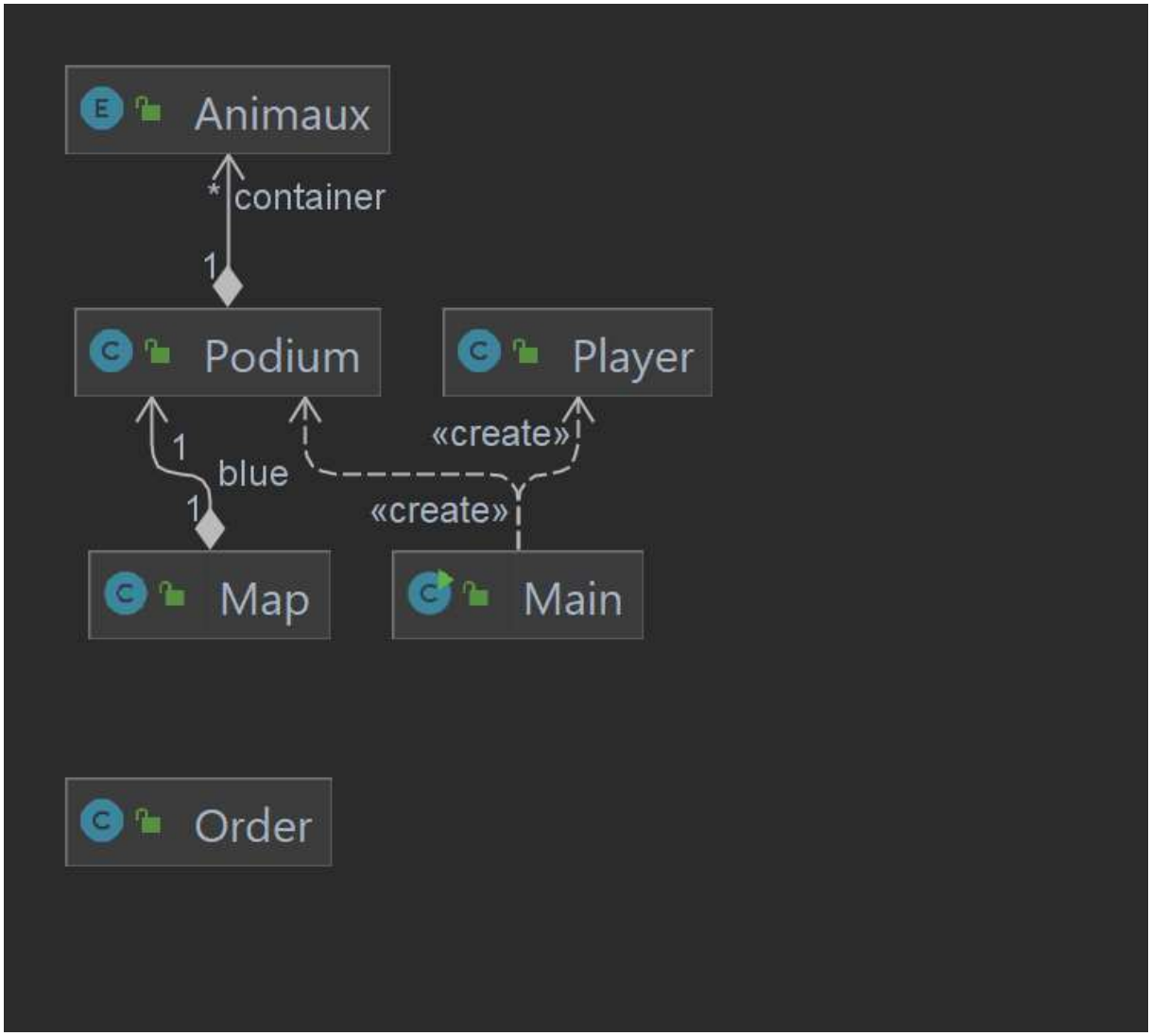
« JOUEUR est le grand gagnant avec SCORE point(s)

Les recettes

Organisation des tests

Afin de vérifier que notre application fonctionne correctement, nous avons réaliser des tests de chaque commandes une à une et effectuer de nombreuses parties en faisant exprès de nous tromper pour vérifier tous les cas d'utilisations possibles. Nous avons aussi intégré à notre programme de nombreux messages d'erreurs pour une meilleure expérience utilisateur.

Diagramme UML



Bilan

Points à améliorer

Nous pensons pouvoir encore optimiser notre algorithme de sélection des cartes, et créer une vraie interface graphique via Swing. Le réel problème que nous avons eu sur ce dernier point et la partie graphique (Créer des rendus en image), cependant nous pensons que si nous avions pu gérer cette partie notre projet Crazy Circus aurait été un vrai jeu vidéo !

Points positifs

Ce projet à été assez sympa à développer. Il nous a permis de découvrir de nouvelles technologies (JavaFX et Swing) à nos dépends 😕. Nous avons aussi découvert comment créer des diagrammes UML sur intellij et éclipse.

Conclusion

Malgré le problème que Nicolas a eu avec JavaFX (corruption du projet), Crazy Circus était un jeu agréable à développer même si nous aurions aimé vous présenter une version ultime du jeu (même si celle que vous avez sous vos yeux est déjà très bien).